101 – Introduction

NoSQL

WIK-NOSQL-101

Intervenant : Jeremy Trufier <jeremy@wikodit.fr>

ynov

wikodit

Not Only SQL

Qu'est-ce donc ?

- Avant tout : une ouverture d'esprit
- SGBD
- Le SQL ne répond pas à tous les besoins
- Problématique du BigData
- Problématique du clustering

Le débat

- Est-ce que NoSQL contient SQL ?
- La différence entre le terme, et la famille admise
- NoSQL englobe des bases relationnelles et non relationnelles

Les acteurs du mouvement

- Google avec BigTable
- Facebook avec Cassandra
- Amazon avec Dynamo

BIG DATA

Les solutions aux problématiques SQL

Problématiques SQL

- Dépendance au schéma
- Mécanismes de réplication Master/Slave
- Difficilement scalable
- Puissant mais language de query complexe et jamais totalement exploité

NoSQL : Le schéma

- Base de donnée SchemaLess
- Schéma géré sur le layer applicatif
- Migration des données maitrisée
- Disponibilité des données

NoSQL : Scalabilité

- Scalabilité horizontale
- Nodes
- Scalabilité automatique

NoSQL : Sharding

- Découpage d'une base en morceau
- Scaling horizontal non répliqué
- Fonctionnement par tag
- Possible de séparer les données dans plusieurs pays, afin de fournir des accès rapides en local

NoSQL : Réplication

- Scaling horizontal
- Généralement pas de Master ni de Slave
- L'écriture peut se faire sur n'importe quel serveur, et est ensuite répliquée ou shardée
- LoadBalancing read/write et répartition de charge

NoSQL : Optimisation

- Requêtes simplifiées et optimisée (enfer du JOIN en SQL)
- Stockage de structure simples ou complexes
- Généralement très rapide
- Supporte généralement les streams

Mais la plupart du temps...

- Les query complexes sont difficiles ou impossibles
- Pas de notion de transactions
- Pas de "Referential Integrity"
- Pas de "Strong Consistency"

Au final

L'objectif de NoSQL est de proposer une solution de stockage de données, la plus proche possible des besoins.*

* SQL inclus

Les principaux types

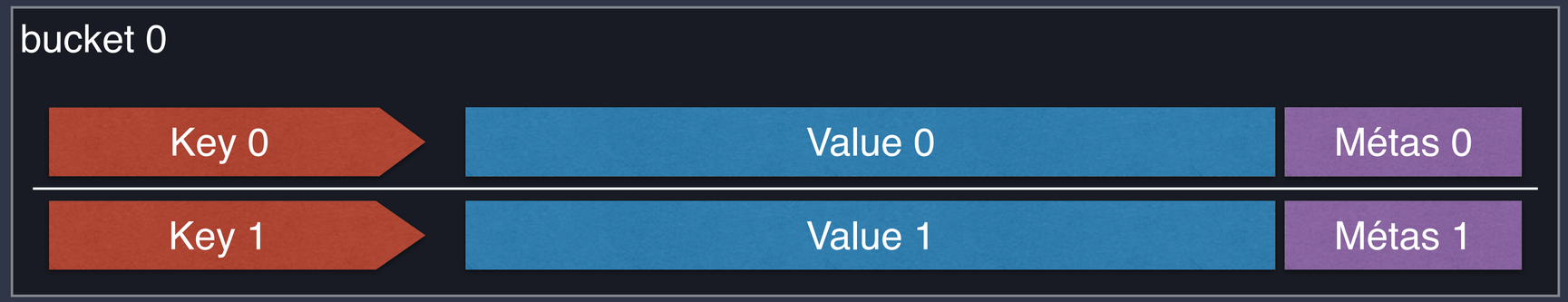
Orienté clé/valeur

Les principaux types

L'essentiel

- Enregistrement = clé + valeur
- Clé est unique
- Valeurs sérialisée
- Pas de table, mais des buckets
- Très utilisé pour le cache

Exemple



key	value	expire
accessToken/ 4ef4c6780bb548c3bbea9 4c60ea50514c	{ "userId": "d699da24-824a-4e68-b2d4-369f2dca842b", "callCount": 213, "lastIp": 81.72.63.54 }	2016-10-10T10:00:00Z
connectedUsers	100	2016-10-10T10:00:00Z
test	toto	2016-10-10T10:00:00Z

Les avantages / inconvénients

+ Les meilleures performances lecture/écriture
+ Utilise très peu d'espace
+ Stocké en mémoire vive (généralement)
+ Très scalable
+ Très simple de mise en place
+ Généralement possible de définir une date d'expiration de la valeur

- Impossible d'extraire en fonction de la valeur (ou très lent)
- Pour lire, il est nécessaire d'avoir la clé (ou une partie de la clé)
- Update multiple difficile

Pour quels besoins ?

- Cache
- Performances
- Pas besoin de relations
- Une simple clé est suffisante pour les requêtes, c'est à dire, pour filtrer ou trier ce n'est pas la bonne solution.

Quelques bases

- Redis*
- memcached*
- OrientDB*
- ArangoDB*
- Riak*
- Berkeley DB*
- *Azure tablestore*
- *Oracle NoSQL*
- ...

* OpenSource

Orienté colonne

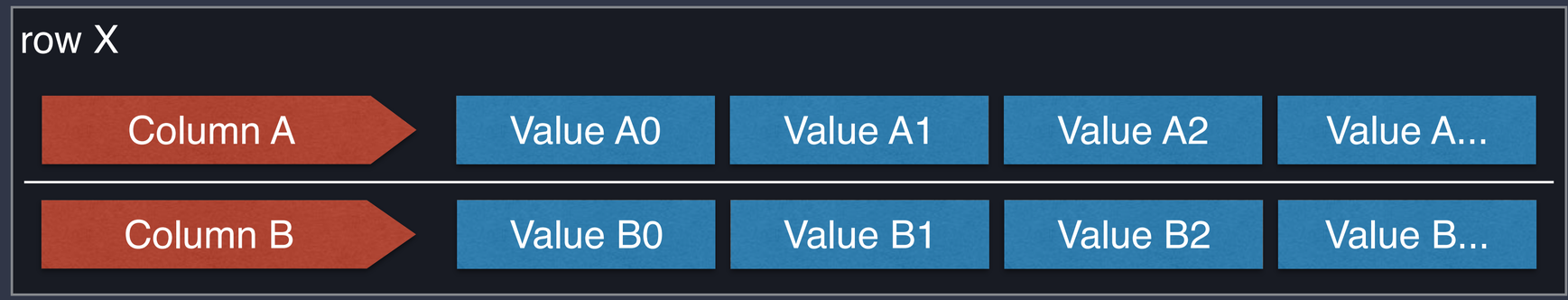
Les principaux types

L'essentiel

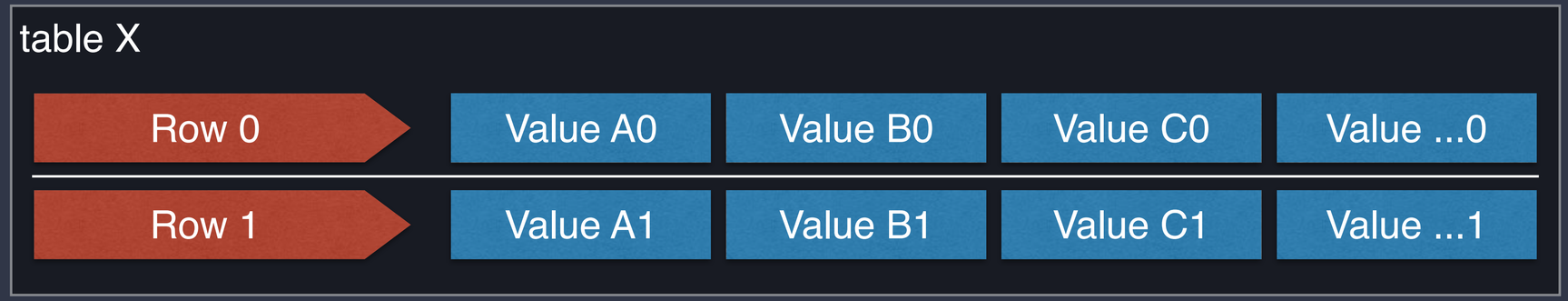
- Chaque colonne est traité individuellement
- Array-processing (opération sur toutes les valeurs d'une colonne)
- "row output" si besoin
- Les données des colonnes sont stockées séparément

Orienté colonne

Exemple



En orienté ligne (comme SGBD SQL traditionnels) :



Exemple

En Orienté Ligne

SELECT AVG(friendCount) WHERE createdAt < 2009-01-01

rowId	pseudo	firstname	lastname	country	friendCount	createdAt
0	Tronix117	Jeremy	Tronix	France	250	2006-05-04
1	Sebvita	Sebastien	Vita	France	500	2007-09-01
3	Nico	Nicolas	Jouffreau	France	10000	2009-10-13
...	...	...	...	...	...	

Exemple

En Orienté Colonne

SELECT AVG(friendCount) WHERE createdAt < 2009-01-01

rowId	pseudo	firstname	lastname	country	friendCount	createdAt
0	Tronix117	Jeremy	Tronix	France	250	2006-05-04
1	Sebvita	Sebastien	Vita	France	500	2007-09-01
3	Nico	Nicolas	Jouffreau	France	10000	2009-10-13
...	...	...	...	...	...	

On a ici parcouru beaucoup moins de données sur disque pour obtenir un résultat

Les avantages / inconvénients

+ Très performant dès lors que toutes les colonnes ne sont pas necessaires
+ Lecture et opération très rapide sur toutes les valeurs d'une colonne
+ Reduction des I/O disques
+ Ajouts, modification, suppression de colonnes simple et rapide
+ Compression très efficace

- Transactionnel lent(si présente)
- Écritures lentes
- Plus lent que SQL pour accéder à des lignes entières selon des colonnes indexées (ex: selection par PK)

Pour quels besoins ?

- Analyse rapide
- Statistiques
- Performances de filtrage et de tri analytique
- Big Data
- Très utilisé dans la finance
- Données similaires dans les colonnes (compression)

Quelques bases

- Hypertable*
- HBase*
- Druid*
- Vertica*
- Cassandra
- *Google BigTable*
- *Amazon AWS Redshift*
- *MS SQL Server 2012*
- *Oracle Database (option In-Memory)*
- ...

* OpenSource

Orienté document

Les principaux types

L'essentiel

- Stockage par documents
- Document = structure flexible (JSON, BSON, ...)
- Document a des métas-données : collection, tags, ...
- Filtre et tri sur le contenu
- Pas de schéma (schema-less)
- Relationnel (ou plutôt référentiel)

Exemple


bdd
Key 0
Name
Type
Value
Name
Type
Value
Key 1
Name
Type
Value
Name
Type
Value
Name
Type
Name
Type
Value
Name
Type
Value

Exemple

key	document	collection
6adc91c6-c218-4b73-a52c-515a4a70a5fd	{ "email": "jean.bon@i.am.a.bad.spy.gouv.fr", "infos": { "firstname": "James", "lastname": "Bond", "alias": "007 }, "adresses": [<d699da24-824a-4e68-b2d4-369f2dca842b>, <261592e2-cecf-437e-80b1-9a447dab7f6a>] }	**users**
d699da24-824a-4e68-b2d4-369f2dca842b	{ "address": "100 rue du Bridge", "city": "Bordeaux" }	**address**
261592e2-cecf-437e-80b1-9a447dab7f6a	{ "address": "5012 paris street", "city": "London", "country": "United Kingdom" }	**address**

Les avantages / inconvénients

+ Réplication
+ Références
+ Read/Write
+ Indexation (+ indexation géospatiale)
+ Flexible (documents mutables)
+ Scalable (sharding)

- Pas de jointures
- Query basiques
- Filtrage applicatif additionnel pour les cas complexes

Pour quels besoins ?

- Back-end avec de gros volumes read/write
- Back-end familiers avec du JSON/BSON
- Besoin de structures nested
- Besoin de flexibilité des données

Quelques bases

- MongoDB*
- CouchDB*
- CouchBase*
- SimpleDB*
- RethinkDB*
- OrientDB*
- ArangoDB*
- ...

* OpenSource

BDD Graphes

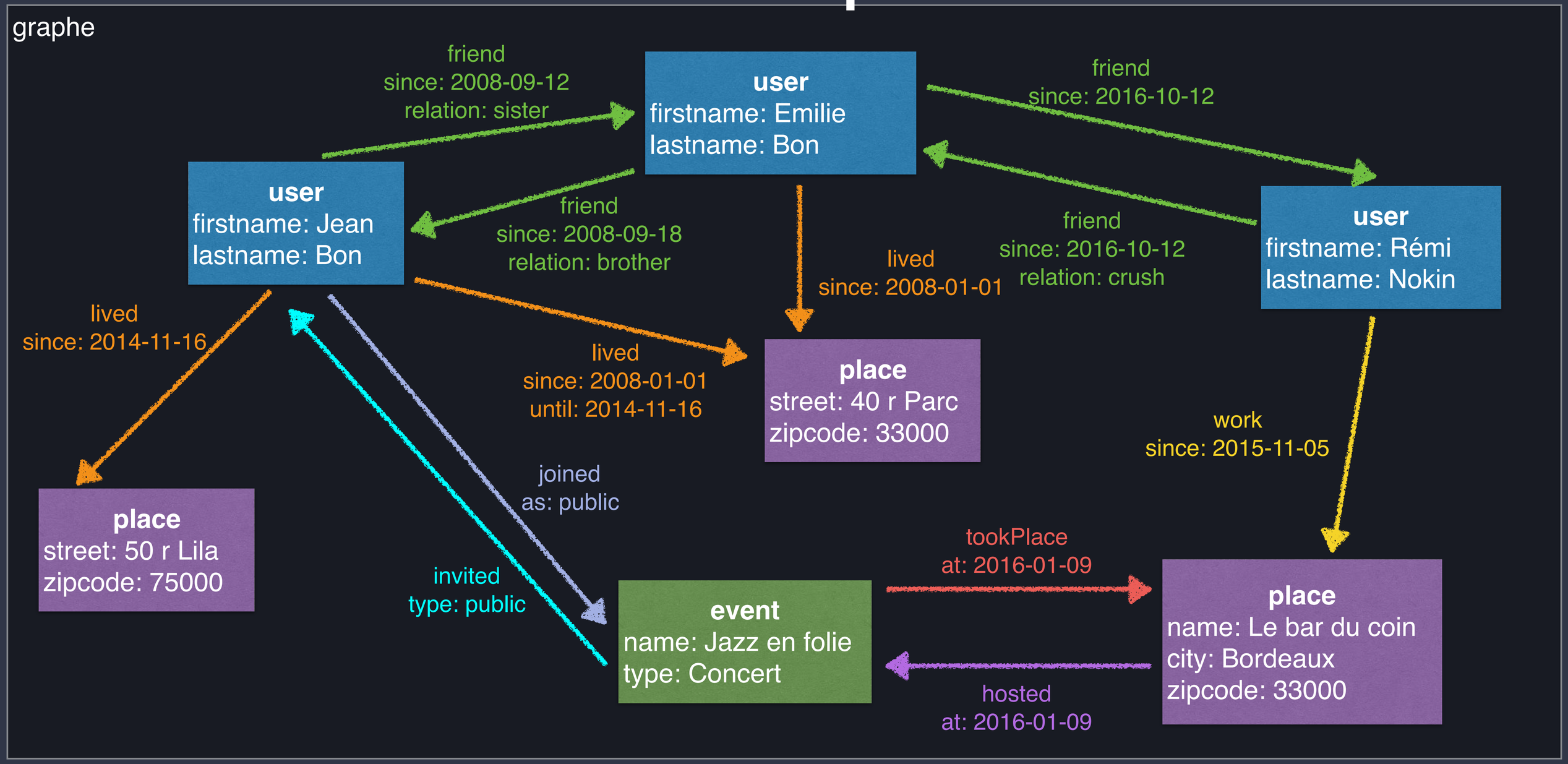
Les principaux types

L'essentiel

- L'ancêtre des BDD relationnelles
- Constitués de noeuds (node), de relations (edge) et de propriétés
- Chaque noeud et chaque relation peuvent avoir des propriétés
- Fonctionnement par voisinage
- Accès extrèmement rapide à des données très éloignées (ex: amis des amis qui aiment la même chose que moi)
- Application de la théorie des graphes

BDD Graphes

Exemple



Les avantages / inconvénients

+ Accès rapide à des noeuds distants
+ Recherches complexes
+ Compression extreme (ex: représentation sous forme de bitmap)
+ Transactionnel

- Statistiques et array processing sur des colonnes impossible ou très lent
- Cohérence et scalabilité compliquée

Pour quels besoins ?

- Besoins de relations sur différents niveaux
- Lorsque des instances d'entités peuvent être reliées à d'autres instances d'entités (ex: le réseau routier)
- Réseaux sociaux !
- Recommandations sur du e-shoping
- Infrastructure systèmes et réseaux
- Intelligences Artificielles (ex: Akinator)

Quelques bases

- Neo4j
- OrientDB
- ArangoDB
- Facebook BDD
- Oracle Spatial And Graph
- ...

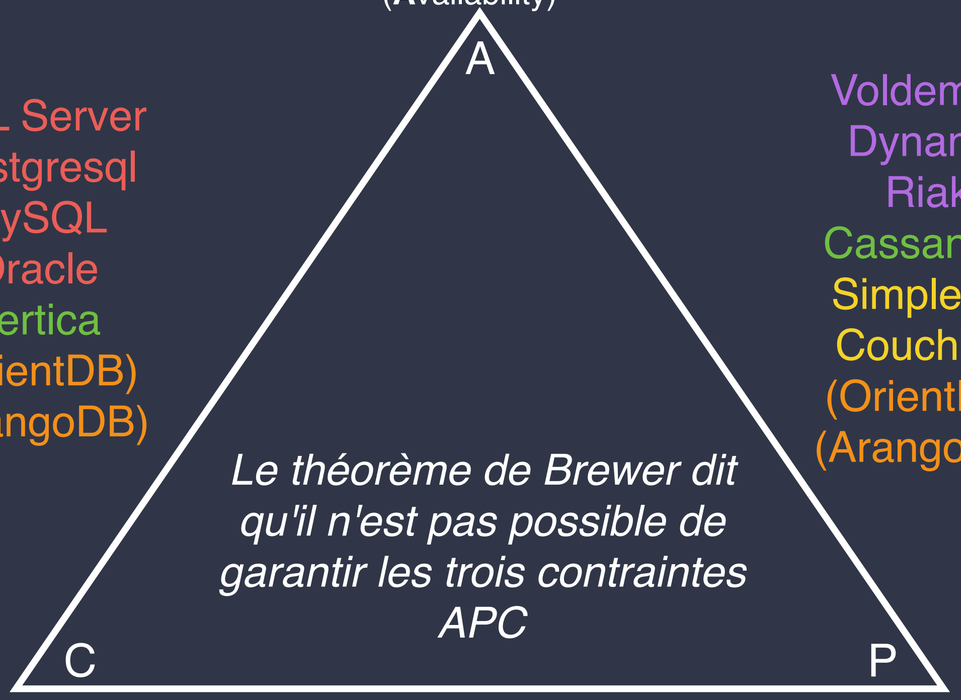
Et en plus simple ?

DISPONIBILITÉ

Lecture et écriture à tout moment
(**A**vailability)

SQL Server
Postgresql
MySQL
Oracle
Vertica
(OrientDB)
(ArangoDB)

Voldemort
Dynamo
Riak
Cassandra
SimpleDB
CouchDB
(OrientDB)
(ArangoDB)



Le théorème de Brewer dit qu'il n'est pas possible de garantir les trois contraintes APC

COHÉRENCE

Les données sont les mêmes pour tous
(**C**onsistency)

TOLÉRANCE AU PARTITIONNEMENT

Aucune coupure, même en cas de soucis réseaux
(**P**artition Tolerance)

Relationnel
Orienté clé/valeur
Orienté colonne
Orienté document
Multi-orientation

BigTable
HyperTable
HBase
MongoDB

Berkeley DB
Memcached
Redis

Source: http://blog.nahurst.com/visual-guide-to-nosql-systems

Félicitations !!

Cours WIK-NOSQL-101 burned :)